

# Kotlin - Break and Continue

## Kotlin Break Statement

Kotlin **break** statement is used to come out of a loop once a certain condition is met. This loop could be a **for**, **while** or **do...while** loop.

### Syntax

Let's check the syntax to terminate various types of loop to come out of them:

```
// Using break in for loop
for (...) {
    if(test){
        break
    }
}

// Using break in while loop
while (condition) {
    if(test){
        break
    }
}

// Using break in do...while loop
do {
    if(test){
        break
    }
}while(condition)
```

If **test** expression is evaluated to true, **break** statment is executed which terminates the loop and program continues to execute just after the loop statment.

### Example

Following is an example where the while loop terminates when counter variable **i** value becomes 3:

```
fun main(args: Array<String>) {
    var i = 0;
    while (i++ < 100) {
        println(i)
        if( i == 3 ){
            break
        }
    }
}
```

```
}  
}
```

When you run the above Kotlin program, it will generate the following output:

```
1  
2  
3
```

## Kotlin Labeled Break Statement

Kotlin **label** is the form of identifier followed by the **@** sign, for example test@ or outer@. To make any Kotlin Expression as labeled one, we just need to put a label in front of the expression.

```
outerLoop@ for (i in 1..100) {  
    // ...  
}
```

Kotlin **labeled break** statement is used to terminate the specific loop. This is done by using break expression with @ sign followed by label name (break@LabelName).

```
fun main(args: Array<String>) {  
    outerLoop@ for (i in 1..3) {  
        innerLoop@ for (j in 1..3) {  
            println("i = $i and j = $j")  
            if (i == 2){  
                break@outerLoop  
            }  
        }  
    }  
}
```

When you run the above Kotlin program, it will generate the following output:

```
i = 1 and j = 1  
i = 1 and j = 2  
i = 1 and j = 3  
i = 2 and j = 1
```

## Kotlin Continue Statement

The Kotlin **continue** statement breaks the loop iteration in between (skips the part next to the continue statement till end of the loop) and continues with the next iteration in the loop.

### Syntax

Let's check the syntax to terminate various types of loop to come out of them:

```
// Using continue in for loop
for (...) {
    if(test){
        continue
    }
}

// Using continue in while loop
while (condition) {
    if(test){
        continue
    }
}

// Using continue in do...while loop
do {
    if(test){
        continue
    }
}while(condition)
```

If **test** expression is evaluated to true, **continue** statement is executed which skips remaining part of the loop and jump to the next iteration of the loop statement.

## Example

Following is an example where the while loop skips printing variable **i** when its value is 3:

```
fun main(args: Array<String>) {
    var i = 0;
    while (i++ < 6) {
        if( i == 3 ){
            continue
        }
        println(i)
    }
}
```

When you run the above Kotlin program, it will generate the following output:

```
1
2
4
5
6
```

# Kotlin Labeled Continue Statement

Kotlin **labeled continue** statement is used to skip the part of a specific loop. This is done by using continue expression with @ sign followed by label name (continue@LabelName).

```
fun main(args: Array<String>) {  
    outerLoop@ for (i in 1..3) {  
        innerLoop@ for (j in 1..3) {  
            if (i == 2){  
                continue@outerLoop  
            }  
            println("i = $i and j = $j")  
        }  
    }  
}
```

When you run the above Kotlin program, it will generate the following output:

```
i = 1 and j = 1  
i = 1 and j = 2  
i = 1 and j = 3  
i = 3 and j = 1  
i = 3 and j = 2  
i = 3 and j = 3
```

## Quiz Time (Interview & Exams Preparation)

**Q 1 - What is the purpose of the break statement in Kotlin?**

- A - Break statement is used to come out of the Kotlin program
- B - Break statement is used to debug a Kotlin program
- C - Break statement is used to come out of a loop
- D - None of the above

**Q 2 - What is labeled expression in Kotlin?**

- A - Labeled expression are used to name different expressions in Kotlin
- B - Labeled expression is an identifier followed by the @ sign
- C - Labeled expression gives more control to jump the program execution
- D - All of the above

**Q 3 - We can use continue statement to skip a part of the loop based on certain condition.**

A - True

B - False

mohamedsohel.co.in